Congestion Avoidance using Frame Relay Architecture

A. Mohan* & Dr. P. Senthil Kumar**

*Research Scholar (Anna University, Chennai), Saveetha Engineering College, Chennai, Tamilnadu, INDIA. E-Mail: mohanmail@sify.com
**Professor & Head, Department of Computer Science and Engineering, SKR Engineering College, Chennai, Tamilnadu, INDIA.
E-Mail: drsenthilkumar2010@gmail.com

***Abstract*—**Frame relay is a one of the packet switching network. In frame relay congestion happens sending of huge volume of data in the low latency network and multiple synchronized servers. Since internet application by produce traffic pattern, due to this huge data sending, reduce the performance of internet and increase response time. In this paper discussed about throughput, bandwidth in frame relay network. The existing works in frame relay, congestion avoidance is not sufficient manner for this type of data sending in network. This method uses window adjustment technique in receiver side to avoid congestion and packet loss. We implement the test; produce the good bandwidth and high efficiency, while sending huge volume of data using frame relay networks.

***Keywords*—**Congestion Avoidance, Control Trigger Algorithm, Frame Relay, Packet Loss, Round Trip Time

***Abbreviations*—**Backward Explicit Congestion Notification (BECN), Bandwidth Delay Product (BDP), Forward Explicit Congestion Notification (FECN), Frame Relay Congestion Control Algorithm (FRCCA), Round-Trip Time (RTT), Transport Control Protocol (TCP)

I. INTRODUCTION

AS the de facto reliable transport-layer protocol, Transport Control Protocol (TCP) is widely used on the Internet and generally works well. However, recent studies [Chaum, 2001; Capkun et al., 2003] have shown that TCP does not work well for many-to-one traffic patterns on high-bandwidth, low-latency networks. Congestion occurs when many synchronized servers under the same Gigabit Ethernet switch simultaneously send data to one receiver in parallel. Only after all connections have finished the data transmission can the next round be issued. Thus, these connections are also called barrier-synchronized [Kong & Hong, 2003]. The final performance is determined by the slowest TCP connection, which may suffer from timeout due to packet loss. The performance collapse of these many-to-one TCP connections is called TCP incast congestion. The frame relay networks packet transmission have only detect the congestion, if occurred using FECN, BECN. These two methods do not prevent congestion in frame relay networks. Still packet loss is low percentage (9.33% approximately) [Zhu et al., 2004]. The basic objective of this paper is to improve the efficiency of the network and reduce the packet loss.

The root cause of TCP incast collapse is that the highly bursty traffic of multiple TCP connections overflows the Ethernet switch buffer in a short period of time, causing intense packet loss and thus TCP retransmission and timeouts. Previous solutions focused on either reducing the wait time for packet loss recovery with faster retransmissions [Capkun et al., 2003], or controlling switch buffer occupation to avoid overflow by using ECN and modified TCP on both the sender and receiver sides [Nagle et al., 2004; Guo et al., 2008; Al-Fares et al., 2008].

This paper focuses on avoiding packet loss before incast congestion, which is more appealing than recovery after loss. Of course, recovery schemes can be complementary to congestion avoidance. The smaller the change we make to the existing system, the better. To this end, a solution that modifies only the TCP receiver is preferred over solutions that require switch and router support (such as ECN) and modifications on both the TCP sender and receiver sides. Our idea is to perform incast congestion avoidance at the receiver side by preventing incast congestion.

The receiver side is a natural choice since it knows the throughput of all TCP connections and the available bandwidth. The receiver side can adjust the receive window size of each TCP connection, so the aggregate burstiness of all the synchronized senders are kept under control. We call our design Frame relay congestion control algorithm (FRCCA) congestion Control for frame relay transmission. However, adequately controlling the receive window is challenging. The receive window should be small enough to avoid incast congestion, but also large enough for good performance and other nonincast cases. A well-performing throttling rate for one incast scenario may not be a good fit for other scenarios due to the dynamics of the number of

connections, traffic volume; we first perform congestion avoidance at the system level. We then use the per-flow state to finely tune the receive window of each connection on the receiver side. The technical novelties of this work are as follows:

To perform congestion control on the receiver side,

- We use the available bandwidth on the network interface as a quota to coordinate the receive window increase of all incoming connections.
- Our per-flow congestion control is performed independently of the slotted time of the Round-Trip Time (RTT) of each connection, which is also the control latency in its feedback loop.
- Our receive window adjustment is based on the ratio of the difference between the measured and expected throughput over the expected.

This allows us to estimate the throughput requirements from the sender side and adapt the receiver window accordingly. We also find that live RTT is necessary for throughput estimation as we have observed that TCP RTT in a high-bandwidth low-latency network increases with throughput [Krevat et al., 2007; Phanishayee et al., 2008; Vasudevan et al., 2009], even if link capacity is not reached.

II. REASONS FOR FRAME RELAY CONGESTION

Frame relay congestion happens when the switch buffer overflows because the network pipe is not large enough to contain all TCP packets injected into the network. The capacity of the network pipe is known as Bandwidth Delay Product (BDP). The network delay consists of three parts: 1) the transmission delay and propagation delay, which are relatively constant; 2) the system processing delay at the sender and receiver server, which increases as the window size increases; and 3) the queuing delay at the Ethernet switch. We define the base delay as the RTT observed when the receive window size is one MSS, which covers the first two parts without queuing. Correspondingly, we have a base BDP, which is the network pipe without queuing at the switch. For the incast scenario, the total BDP consists of two parts: the base BDP and the queue size of the output port on the Ethernet switch. Previous work focused on how to mitigate the impact of timeouts, which are caused by a large amount of packet loss on incast congestion.

First, the available bandwidth at the receiver side is the signal for the receiver to perform congestion control. As incast congestion happens at the last hop, the incast receiver should detect such receiving throughput burstiness and control the throughput to avoid potential incast congestion. If the TCP receiver needs to increase the TCP receive window, it should also predict whether there is enough available bandwidth to support the increase. Furthermore, the receive-window increase of all connections should be jointly considered. Second, the frequency of receive-window-based congestion control should be made according to the per-flow feedback loop delay independently. In principle, the congestion control dynamics of one TCP connection can be regarded as a control system, where the feedback delay is the RTT of that TCP connection. When the receive window is adjusted, it takes at least one RTT before the data packets following the newly adjusted receive window arrive. Thus, the control interval should be larger than one RTT, which changes dynamically according to queuing delay and system overhead.

Third, a receive-window-based scheme should adjust the window according to both link congestion status and the application requirements. The receive window should not restrict TCP throughput when there is available bandwidth and should throttle TCP throughput before incast congestion occurs. Consider a scenario where a TCP receive window is increased to a large value but is not decreased after the application requirement is gone. If the application resumes, congestion may occur with a traffic surge in such a large receive window. Therefore, the receiver should differentiate whether a TCP receive window over satisfies the achieved throughput of a TCP connection and, if so, decrease its receive window. Based upon these three observations, our receive-window based incast congestion control is intended to set a proper receive window for all TCP connections sharing the same last hop. Considering that there are many TCP connections sharing the bottlenecked last hop before incast congestion, we adjust the TCP receive window to make those connections share the bandwidth equally. This is because in a data center, parallel TCP connections may belong to the same job, where the last one finished determines the final performance. Note that the fairness controller between TCP flows is independent of the receive window adjustment for incast congestion avoidance, so that any other fairness category such as proportional to weights can be deployed if needed.

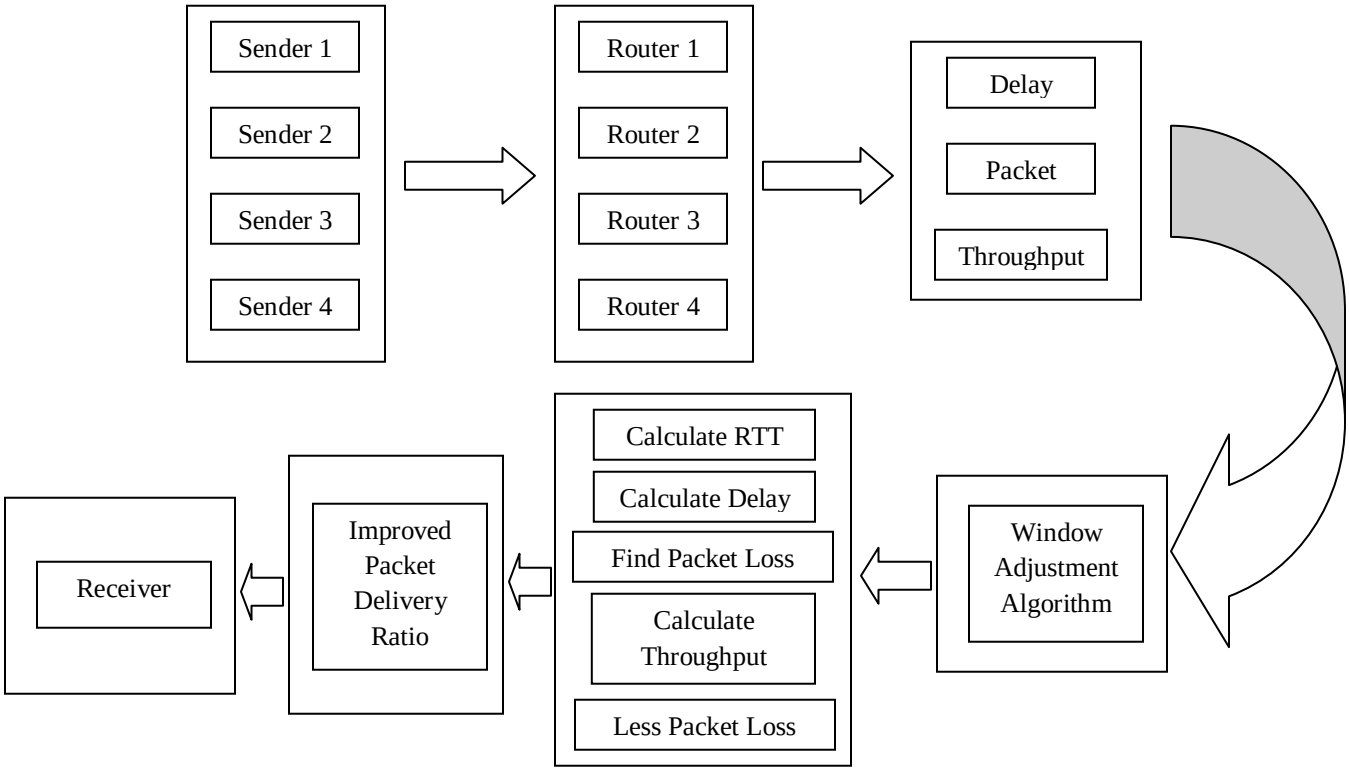

Figure1 – Flow Chart of the Proposed Separation Algorithm

III. FRAME RELAY CONGESTION CONTROL ALGORITHM (FRCCA)

The frame relay network has two mechanisms for detect the congestion in frame relay networks. There are FECN, BECN and Leaky Bucket Algorithm (LBA). The LBA is not suitable for sending huge amount of data in frame relay networks. This algorithm is not more efficient compare with recent techniques. The existing system has high packet loss and network delay.

This is not including in existing work. The work perform avoid congestion in receiver side while sending huge amount of data simultaneously to a receiver. Here generate a algorithm like window adjustment algorithm. This algorithm focus reduces the RTT time between the data sending time and increase the throughput of the network. Since we can avoid the congestion in the frame relay networks. The frame relay network transmits the data as frames. A single packet segment into more number of frames. This frame have unique id and verify the frames at receiver side using the concept of frame check sequence. Router plays major role in frame relay networks. The router route the frame corresponding destination or next router. The router may use different routing algorithm for route the frame to destination. Each and every router has some limited buffer size. The congestion control algorithm works based on window adjustment algorithm. This algorithm concentrates on measure the RTT time between each and every frame transmissions and calculates how much amount of data arrived in particular time period, and then reduces the RTT time between the transmissions. This is helps to increase the network efficiency and reduce the packet loss. This algorithm is suitable only for receiver side congestion control.

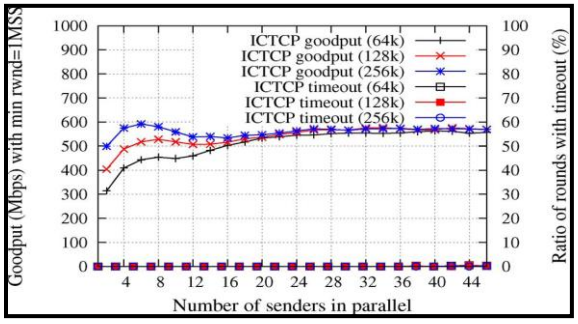

Figure 2 – FRCCA Goodput and Ratio of Experimental Rounds Suffer at least One Timeout with a Minimal Receive Window of 1 mss

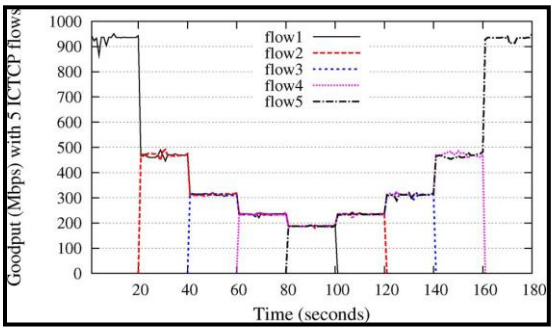

Figure 3 – Goodput of Five ICTCP Flows that Start Sequentially with 20-S Intervals and 100 S Duration, from Five Sending Servers and to the Same Receiving Server under the Same Switch

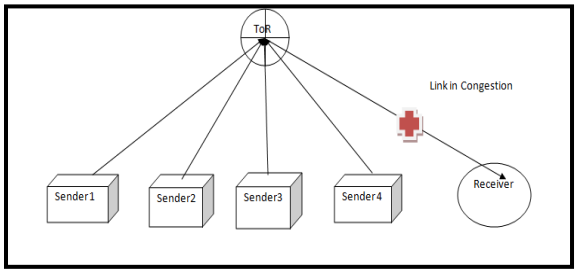

Figure 4 – Line in Congestion

IV. CONCLUSION

In contrast to previous approaches that used a fine-tuned timer for faster retransmission, we focus on a receiver-based congestion control algorithm to prevent packet loss. ICTCP adaptively adjusts the TCP receive window based on the ratio of the difference of achieved and expected per-connection throughputs over expected throughput, as well as the last-hop available bandwidth to the receiver. In frame relay analyses the packet transmission and measure the packet loss and avoid the congestion using FRCCA using simulation.

REFERENCES

[1] D. Chaum (2001), "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", *Communications of the ACM*, Vol. 4, No. 2.

[2] S. Capkun, L. Buttyan & J. Hubaux (2003), "Self-Organized Public-Key Management for Mobile Ad Hoc Networks", *IEEE Transactions on Mobile Computing*, Vol. 2, No. 1, Pp. 52–64.

[3] J. Kong & X. Hong (2003), "ANODR: Aonymous On Demand Routing with Untraceable Routes for Mobile Ad-Hoc Networks", *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Pp. 291–302.

[4] B. Zhu, Z. Wan, F. Bao, R. H. Deng & M. KankanHalli (2004), "Anonymous Secure Routing in Mobile Ad-Hoc Networks", *Proceedings of IEEE Conference on Local Computer Networks*, Pp. 102–108.

[5] D. Nagle, D. Serenyi & A. Matthews (2004), "The Panasas ActiveScale Storage Cluster: Delivering Scalable High Bandwidth Storage", *Proceedings of ACM/IEEE Conference on Supercomputing*, Pp. 53.

[6] E. Krevat, V. Vasudevan, A. Phanishayee, D. Andersen, G. Ganger, G. Gibson & S. Seshan (2007), "On Application-Level Approaches to Avoiding TCP Throughput Collapse in Cluster-based Storage Systems", *Proceedings of Supercomputing*, Pp. 1–4.

[7] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang & S. Lu (2008), "DCell: A Scalable and Fault Tolerant Network Structure for Data Centers", *Proceedings of ACM SIGCOMM*, Pp. 75–86.

[8] M. Al-Fares, A. Loukissas & A. Vahdat (2008), "A Scalable, Commodity Data Center Network Architecture", *Proceedings of ACM SIGCOMM*, Pp. 63–74.

[9] E. Phanishayee, V. Krevat, D. Vasudevan, G. Andersen, G. Ganger, G. Gibson & S. Seshan (2008), "Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems", *Proceedings of USENIX FAST*, Article No. 12.

[10] Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. Andersen, G. Ganger, G. Gibson & B. Mueller (2009), "Safe and Effective Fine-Grained TCP Retransmissions for Datacenter Communication", *Proceedings of ACM SIGCOMM*, Pp. 303–314.

Mohan Annamalai is currently working as Assistant Professor in Saveetha Engineering College, Thandalam, Chennai. I received M.Tech (IT) in 2008 from Sathyabama University, Chennai. I have 11 years of teaching experience in various engineering colleges. Am doing research in Anna University, Chennai. Am very much interested in programming languages and networking. I published 14 papers in national and international conferences.

Senthil Kumar Ponnusamy is currently working as Professor and Head of the Department of Information Technology, SKR Engineering College, Chennai. He received his ME in 2002 from Arulmigu Kalasalingam college of engineering, Krishnankovil and PhD in 2010 from Bharath University, India. He has 13 years of experience in various engineering college. He has published 26 papers in various national and International journal and conferences. His area of research is network.